42qu.com
源码 & 架构 导读

张沈鹏
zuroc.42qu.com
2012.10

简介

42qu.com 是一个开源的 python sns 项目

基于 zweb 框架 , zweb 基于 tornado

代码地址

http://42qu-source.42qu.com/10740410

每个功能都按模块划分

.app/ 下面有所有的模块

base 最基本的模块

- 用户
- 收发邮件
- 上传图片
- ...

auth

- 注册
- 登录
- 找回密码

各模块下的有自己 view , template & model

```
work@v1 ~/42qu/.app/book $ ls -alh
总用量 36K
drwxr-xr-x  9 work work 4.0K  9月 10 14:52 .
drwxr-xr-x 20 work work 4.0K  9月 28 12:01 ..
drwxr-xr-x  2 work work 4.0K  9月 28 12:01 coffee
drwxr-xr-x  3 work work 4.0K  9月 28 12:01 css
drwxr-xr-x  3 work work 4.0K  9月 10 14:52 html
-rw-r--r--  1 work work    0  9月 10 14:52 __init__.py
drwxr-xr-x  2 work work 4.0K  9月 28 12:01 js
drwxr-xr-x  9 work work 4.0K  9月 10 14:52 misc
drwxr-xr-x  2 work work 4.0K  9月 28 12:01 model
drwxr-xr-x  4 work work 4.0K  9月 28 12:01 view
```

好处

网站各个功能模块的代码都物理隔离

方便让不同的人维护

如何解决模块间循环依赖

比如　user 模块 有一个函数

删除 spammer 的所有发言和填写的资料

每个模块都有自己的数据要删除

比如 Blog 模块

Blog 模块依赖于 user，但是 user 不应该依赖于 Blog

如何解决　？

引入 SIGNAL

清除资料的时候 发出 SINGAL

user 模块下
from zweb.signal import SIGNAL
SIGNAL.user_rm.send(user_id)

各个模块 分别拦截此 SINGAL，删除本模块产生的内容
blog 模块下

@SIGNAL.user_rm
def _user_rm(user_id):
    ... ...

Signal 的实现

```
class Signal(object):
    def __init__(self):
        self.receiver = []

    def send(self, *args):
        for func in self.receiver:
            func(*args)

    def __call__(self, func):

    self.receiver.append(func)
```

```
class _(object):
    def __getattr__(self, name):
        d = self.__dict__
        if name not in d:
            d[name] = Signal()
        return d[name]
```

```
SIGNAL = _()
```

Signal 与 消息队列 的 整合

消息队列

可以将一些耗费时间的任务放到后台执行

celery (芹菜)

使用 redis 作为后端 (可以选用多种后端)

使用 msgpack 序列化

Celery 初始化

```
from celery import Celery

CELERY_BROKER_URL =
"redis://127.0.0.1:6379/1"

celery =
Celery(broker=CELERY_BROKER_URL)
celery.conf.CELERY_TASK_SERIALIZER =
'msgpack'
```

Signal 与 消息队列 的 整合

```
@SINGAL.user_rm.delay
def _user_rm(user_id):
    pass
```

Celery

绑定一个函数

```
@celery.task
def test(a,b,c):
    pass
```

同　Celery 异步调用

```
test.delay(1,2,3)
```

整合 celery 和 singal

```
#coding:utf-8
from model._db import celery
from misc.config import HOST

class Signal(object):
    def __init__(self, name):
        self._sync = []
        self._async = []


    @celery.task(name=name)
        def _(*args):
            for func in self._async:
                func(*args)

        self.task = _delay
```

```
def send(self, *args):
    for func in self._sync:
        func(*args)
    if self._async:
        self.task(*args)

def __call__(self, func):
    self._sync.append(func)
    return func

def delay(self, func):
    self._async.append(func)
    return func
```

整合 celery 和 singal

```
class _(object):
    def __getattr__(self, name):
        d = self.__dict__
        if name not in d:
            d[name] = Signal("%s.%s"%(HOST,name))
        return d[name]


SIGNAL = _()
```

celery make

misc/celery/make.py

会扫描 所有包含

@SINGAL.xxx.delay

的文件 ， 自动生成一个导入文件

python make.py

生成 celery_import.py

celery -A celery_import worker --loglevel=info

图片存储

图片存储在又拍云

又拍云的表单 API

可直接上传文件到对方服务器

最简单的表单上传

代码地址

42qu/html/base/_util/form.html

```
<form action="http://v0.api.upyun.com/${bucket}"
method="POST" enctype="multipart/form-data">
    <input name="policy" type="hidden" value="${policy}">
    <input name="signature" type="hidden" value="$
    {signature}">
    ... ... ...
</form>
```

利用 iframe 实现异步上传

http://book.42qu.com/new

利用 iframe 实现异步上传

只有一个表单提交 ， 需要提交数据给后台

参考代码

42qu/.app/book/html/root/new.html

42qu/.app/book/coffee/new.coffee

参考文章

http://blog.leezhong.com/tech/2011/05/06/crossdomain-upload.html

利用 iframe 实现异步上传

一个隐藏的 iframe

```
<IFRAME id="upyun" name="upyun"  src="about:blank"  frameborder='0'>
    </IFRAME>
```

一个普通的表单

```
<%form:upyun name="MeetImg" target="upyun"
   return_url="http://${request.host}/img">
     <input type="file" id="img_input" name="file">
</%form:upyun>
```

一段 coffee script

```
   $('#img_input').change ->
       if this.value
           form.submit()
```

利用 iframe 实现异步上传

```
@route("/img")
class _img(LoginView):
    def get(self):
        code = self.get_argument('code', None)
        if code == '200':
            url = '_'.join([self.get_argument('url')[1:], self.get_argument('image-width'), self.get_argument('image-height')])
            self.finish("""<script>parent.set_img("%s");</script>"""%url)
        else:
            self.finish("""<script>alert(' 上传错误 ')</script>""")
```

swf uploader 多文件上传 & 显示进度

http://zuroc.42qu.com/po/img/new

swf uploader 多文件上传 & 显示进度

参考代码 42qu/.app/po/js/swfpo.js

页面上传递一些参数

```
<script>
UPYUN = [
    "z-img",
    "img.42qu.us",

    "eyJidWNrZXQiOiJ6LWltZyIsImV4cGlyYXRpb24iOjEzNTA3MzM0MDksInNhdmUta2V5IjoiL3tmaWxlbWQ1fSJ9","b8eb8ba0a099076add97fd9b2d4184a8"
]
</script>
```

图片剪裁

简单的缩放 / 剪裁

upyun 可以直接自定义缩略图

复杂的剪裁 大头像 -> 剪裁为小头像

一个独立的服务

https://bitbucket.org/fy0/pic-cdn

图片剪裁　. pic-cdn

http://42tu.us/
　f9043354285968eed35d5e99df9b5d62/
　　239,36,245,245!96

可以 配合　nginx proxy cache
和
免费 CDN
http://www.webluker.com
一起使用

这样做的好处

核心代码中不涉及

图片处理 / 文件存储

的逻辑

不需要考虑如何备份文件

节省流量

上传 & 浏览都不经过我的服务器

图片实现了 CDN 加速

静态文件的引用 CSS/JS

js/_hash_.py

css/_hash_.py

```
__HASH__ = {
    "book/drag.js" : 'ewey1KPg5EXq7q9TzeyDDQ.js', #book_drag
    "vps/vm_reboot.js" : 'K44cPFCcqXD6bBBO5L-qhg.js', #vps_vm_reboot
    "po/reply.js" : '-rC3KwicOSF3VWYZmYD-mw.js', #po_reply
    "lnav/notify.js" : '79_kj4GP3DeXksSySJFxRQ.js', #lnav_notify
    "base/popreply.js" : '_s9LVaizQakqXQPBlSQ6Xg.js', #base_popreply
    "star/god/tag/index.js" : 'gRDTJuLN3QdvMzKjprZDaQ.js', #star_god_tag_index
    "vps/edit.js" : '0h618hYq--ZvUUT_sMs9og.js', #vps_edit
```

静态文件的引用 CSS/JS

模版里面的代码

```
<link rel="stylesheet" href="${css.init|n}" type="text/css">
<script src="${js.init|n}"></script>
<link rel="stylesheet" href="${css.auth_init|n}"
    type="text/css">
<script type="text/javascript" src="${js.auth_init|
    n}"></script>
```

线上服务器 . 静态文件的引用

是压缩过的 css 和 js

```
<link rel="stylesheet"
href="http://s.42qu.net/eGkr5SoySMXpMBQr7xKLmQ.css"
type="text/css">

<script
src="http://s.42qu.net/_hwLCxvy7fJNaGab4H6hBg
    .js">
</script>
```

merge.conf

无论 css 还是 js，每个目录下都可以有 merge.conf
把零散的 js / css 合并

work@vps372 ~/42qu $ cat js/merge.conf
init.js :
    const.js
    lib/jquery.js
    lib/fancybox.js
     lib/jquery_ext.js
...

开发模式下 ，直接使用零散的小文件

```
function LOAD(js){ document.write('<script src="'+js+'"></'+"script>")
}
LOAD('http://dev-jss.realfex.tk/const.js')
LOAD('http://dev-jss.realfex.tk/lib/jquery.js')
LOAD('http://dev-jss.realfex.tk/lib/fancybox.js')
LOAD('http://dev-jss.realfex.tk/lib/jquery_ext.js')
LOAD('http://dev-jss.realfex.tk/lib/cookie.js')
LOAD('http://dev-jss.realfex.tk/lib/post_json.js')
LOAD('http://dev-jss.realfex.tk/lib/upload.js')
LOAD('http://dev-jss.realfex.tk/lib/textarea_elastic.js')
LOAD('http://dev-jss.realfex.tk/lib/util.js')
LOAD('http://dev-jss.realfex.tk/lib/suffix.js')
LOAD('http://dev-jss.realfex.tk/lib/jquery.input.js')
```

剥离不重要的服务到外部

短网址 -> SAE

网站出现故障不能访问的时候
短网址服务应该还是正常工作

本地存储短网址 ，方便实时生成

SAE 做反向代理 ，存储并记录下 短网址 -> 真实网址的映射

RSS 同步服务 in dotcloud.com

代码 ：https://bitbucket.org/MichaelGe/longrss

dotcloud 类似 SAE

但是

1. 支持 ssh 到服务器上 (比如配置个 crontab 什么的)
2. 有 redis mangodb 等等众多选择
3. 在国外， 可以抓取一些被墙的 RSS

其他用到的技术

coffee script

zorm

redis + lua

thrift